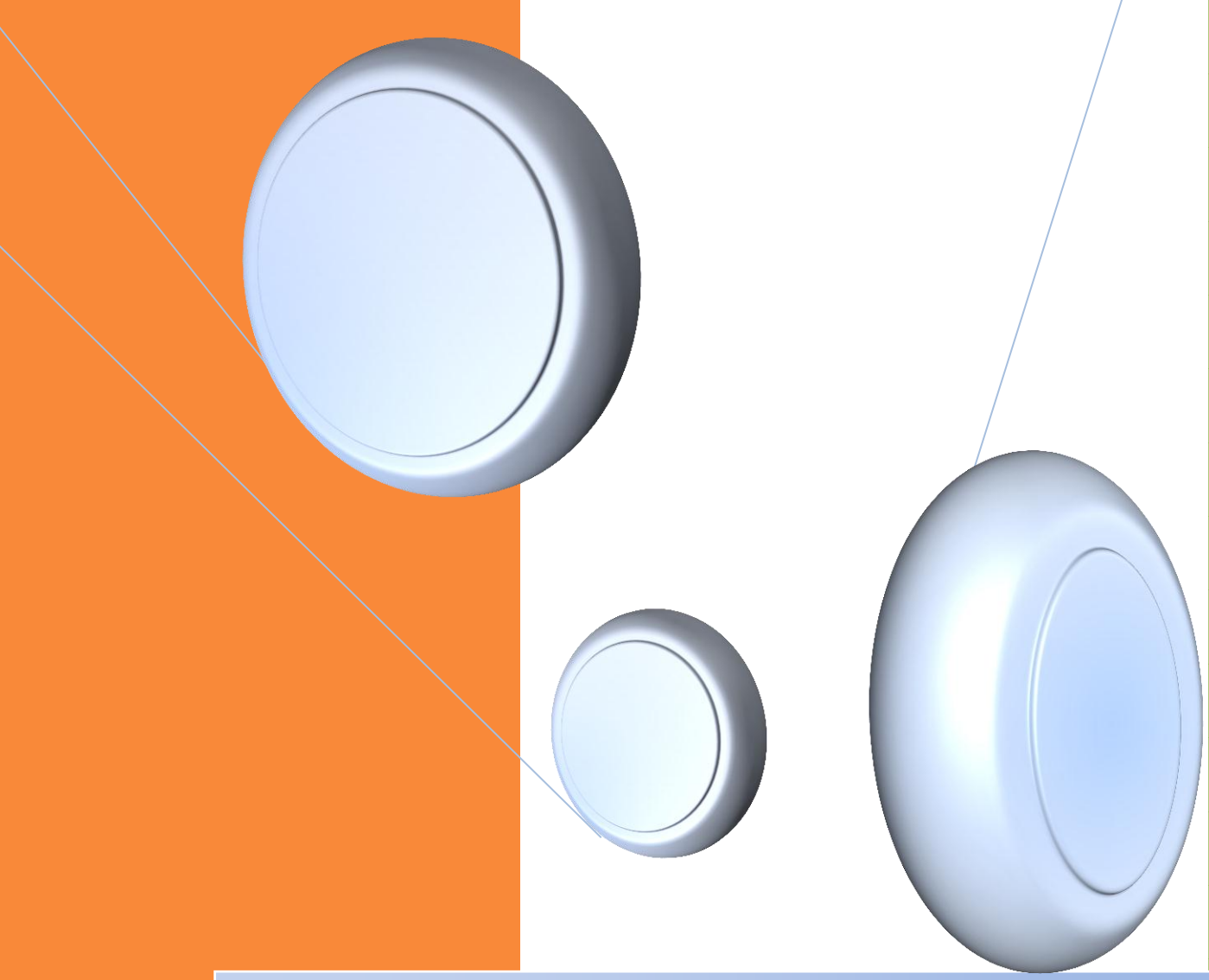


OceanBase 数据库
安装指南

版本：01

日期：2013.7.30

前 言

概述

本文档主要介绍OceanBase数据库的安装流程和安装方法，可以帮助安装工程师完成OceanBase数据库的安装。

读者对象

本文档主要适用于：

- 安装工程师。
- 数据库管理工程师。

修订记录

修改记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本。

目 录

1 安装前须知

介绍了安装 OceanBase 前您需要了解的基本信息。

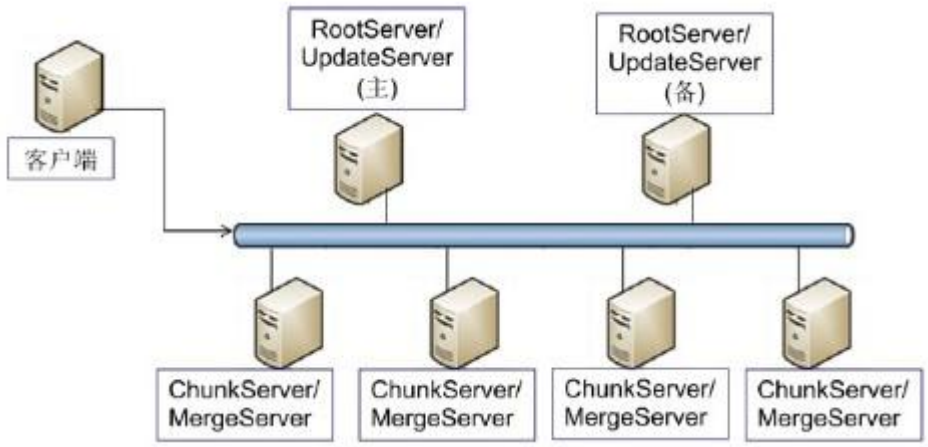
1.1 产品简介

OceanBase 数据库是阿里集团研发的可扩展的分布式关系数据库，实现了数千亿条记录、数百淘宝数据上的跨行跨表事务，主要支持收藏夹、直通车报表、天猫评价等 OLTP 和 OLAP 在线业务。

OceanBase 的数据主要可以分为基准数据和增量数据。基准数据是只读数据，增量数据是需要修改更新的数据。OceanBase 数据库内部通过合并操作定期将增量数据融合到基准数据中。

OceanBase 数据库组网如图 1-1 所示。

图 1-1 OceanBase 数据库组网



- Root Server
  主控服务器，主要进行集群管理、数据分布和副本管理。
- Update Server
  更新服务器，是集群中唯一能够接受写入的模块，存储每日更新的增量数据。
- Chunk Server
  基准数据服务器，存储 OceanBase 数据库中的基准数据，提供数据读取服务、执行定期合并以及数据分发。
- Merge Server
  合并服务器，主要提供协议解析、SQL 解析、请求转发、结果合并和多

表操作等功能。
Listener 是 OceanBase 集群内部特殊的 Merge Server，只负责从集群的内部表中查询主备集群的流量分布信息和所有的其他 Merge Server 的地址列表。

- 客户端
  客户端中存放了多个集群的 Root Server 地址列表，并根据集群的流量分配比例将读写操作发往不同的集群。

1.2 部署模式

OceanBase 数据库部署模式灵活，可满足用户多种需求。

OceanBase 部署模式说明如**表 1-1** 所示。

表 1-1 部署模式

部署模式	说明
单机部署	Root Server、Update Server、Chunk Server 和 Merge Server 部署在同一台服务器上。
分开部署	Root Server、Update Server、Chunk Server 和 Merge Server 部署在不同服务器上。 • Root Server 可采用主备双机模式。采用主备双机模式时，需要先安装 HA，详细请参见《线上 HA 自动 BUILD 工具》。 建议与 Update Server 合设。 • Update Server 可采用主备双机模式。主备由 Root Server 决定。 建议与 Root Server 合设。 • Chunk Server 存储 OceanBase 数据库的基准数据。基准数据一般存储两份或者三份，可配置。可根据需求部署多台。 • Merge Server 对 Update Server 上的动态数据和 Chunk Server 上的静态数据进行合并。可根据需求部署多台。 Listener 是特殊的 Merge Server，在 Root Server 上提供服务。

部署模式	说明
集群部署	OceanBase 集群，即每个 OceanBase 中均含有 Root Server、Update Server、Chunk Server 和 Merge Server 服务。 在初始化 OceanBase 时指定主备。

在不同的部署模式下，OceanBase 的安装方式相同，启动方式不同。

例如，现需部署服务器 A 为 Root Server、Update Server；服务器 B 为 Chunk Server 和 Merge Server。只需在服务 A 和服务器 B 中分别安装 OceanBase，然后在服务器 A 中启动 Root Server、Update Server；服务器 B 中启动 Chunk Server 和 Merge Server。

本文档主要通过部署单机介绍 OceanBase 数据库的安装方法。

1.3 软硬件要求

OceanBase 数据库服务器最低配置要求如下：

- Root Server、Update Server、Chunk Server 和 Merge Server 合设时，采用 64 为 Linux 系统，内存 10GB，磁盘空间 100M。
- Root Server、Update Server、Chunk Server 和 Merge Server 分开部署时如表 1-2 所示。

表 1-2 服务器配置

模块	操作系统	内存	磁盘空间
Root Server/ Listener	Linux，64-bit	1GB	20GB
Update Server	Linux，64-bit	3GB	20GB
Chunk Server	Linux，64-bit	1GB	20GB
Merge Server	Linux，64-bit	1GB	20GB

1.4 安装规划

安装规划主要包括服务器规划、目录规划和磁盘挂载点规划。

1.4.1 服务器规划

OceanBase 数据库服务器规划如表 1-3 所示。

表 1-3 服务器规划

规划项	规划
服务器 IP	10.10.10.2
端口	• Root Server：服务端口 3500。 • Update Server：服务端口 2700，合并操作端口 2710。 • Chunk Server：服务端口 2600。 • Merge Server：服务端口 2800，MySQL 协议端口 3030。 • Listener：服务端口 2828，请勿修改。
安装用户	admin ***注意：****采用 RPM 安装时，安装用户必须为“admin”。*
用户密码	Abc@123
安装目录	/home/admin/oceanbase
集群 ID	1
App 名称	obtest

1.4.2 目录规划

OceanBase 各 Server 的数据目录规划如表 1-4 所示。

表 1-4 目录规划

规划项	规划
Root Server	• 数据目录：/home/admin/oceanbase/data/rs • 日志目录：/home/admin/oceanbase/data/rs_commitlog
Chunk Server	数据目录： • /home/admin/oceanbase/data/1 • /home/admin/oceanbase/data/2 • ..... • /home/admin/oceanbase/data/10

规划项	规划
Update Server	数据目录： • /home/admin/oceanbase/data/ups_data/raid0/store0 • /home/admin/oceanbase/data/ups_data/raid0/store1 • /home/admin/oceanbase/data/ups_data/raid1/store0 • /home/admin/oceanbase/data/ups_data/raid1/store1 日志目录： /home/admin/oceanbase/data/ups_commitlog

1.4.3 磁盘挂载点规划

OceanBase 的 Chunk Server 和 Update Server 分别需要存储静态数据和动态数据，建议使用单独的磁盘进行数据存储。磁盘挂载点的规划如表 1-5 所示。

表 1-5 磁盘挂载点规划

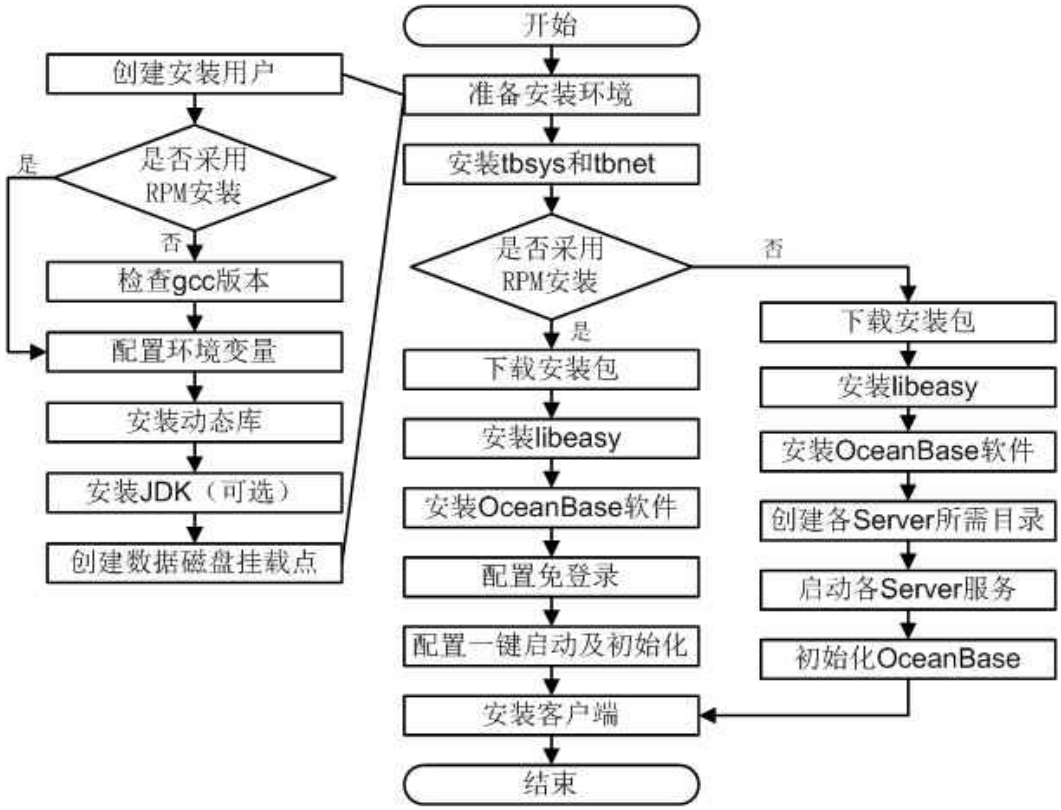
规划项	规划
Chunk Server	10 块物理磁盘或虚拟磁盘： • /data/1 • /data/2 • - ..... • - /data/10
Update Server	数据存放磁盘（4 块 SSD 盘或一块 PCI-E Flash 卡）： • /data/1 • /data/2 • /data/3 • /data/4

1.5 安装流程

主要介绍 OceanBase 数据库的安装流程，有助于您更好地完成安装任务。

OceanBase 数据库安装流程如图 1-2 所示。

图 1-2 安装流程


开始
创建安装用户
准备安装环境
是否采用
RPM安装
是
否
检查gcc版本
配置环境变量
安装动态库
安装JDK（可选）
创建数据磁盘挂载点
安装tbsys和tbnet
是否采用
RPM安装
否
是
下载安装包
安装libeasy
安装OceanBase软件
配置免登录
配置一键启动及初始化
下载安装包
安装libeasy
安装OceanBase软件
创建各Server所需目录
启动各Server服务
初始化OceanBase
安装客户端
结束

2 准备安装环境

准备安装环境主要包括创建安装用户、检查 gcc 版本、配置环境变量、安装动态库、安装 JDK 和创建数据磁盘挂载点。

2.1 创建安装用户

创建 OceanBase 安装用户的操作步骤如下：

1. 以 **root** 用户登录 OceanBase 服务器。
2. 执行如下命令，创建 OceanBase 的安装用户。
   useradd -d /home/admin -s /bin/bash -m admin
3. 执行如下命令，为用户“admin”设置密码。
   passwd admin
4. 您需要根据系统的提示输入两次密码“Abc@123”。
5. 为“obuser”赋予“sudo”权限。
   a. 执行以下命令，添加“/etc/sudoers”文件的写权限。
      chmod u+w /etc/sudoers
   b. 使用 **vi** 编辑器，在“/etc/sudoers”文件中“root ALL=（ALL） ALL”后添加语句，如黑体部分所示。

```
root ALL=（ALL） ALL



admin ALL=(ALL) ALL
```

   c.执行以下命令，删除“/etc/sudoers”文件的写权限。
      chmod u-w /etc/sudoers

2.2 检查 gcc 版本

如果您采用 RPM 安装，则可以跳过本小节。在采用源码安装 OceanBase 前，确认 gcc 版本为 4.1.2，否则会造成编译失败。

检查 gcc 版本的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行 **gcc --version** 命令，检查 gcc 版本，系统显示如下。
   如果您的 gcc 版本不是 4.1.2，请参考“7.2 安装 gcc 4.1.2”中的内容，

安装 gcc 的 4.1.2 版本。

```
gcc (GCC) 4.1.2 20080704 (Red Hat 4.1.2-51)
Copyright (C) 2006 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO  warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

2.3 配置环境变量

OceanBase 在运行时需要使用到动态库，因此安装 OceanBase 前需要配置环境变量，操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 用 **vi** 编辑器在“/home/admin/.bashrc”文件中，添加如下语句：
   说明：*拷贝添加以下语句时，请删除#后的注释语句。其中“/home/admin/ocaenbase”为安装目录。*

```
#set taobao lib Environment Variables:
export TBLIB_ROOT=~/tb-common-utils
export
LD_LIBRARY_PATH=/home/admin/oceanbase/lib:/usr/local/lib/libsnappy.so:/usr/:/usr/lib/:/usr/local/lib:/lib:$TBLIB_ROOT/lib
#set Lib easy Environment Variables:
export EASY_ROOT=$TBLIB_ROOT
export EASY_LIB_PATH=$EASY_ROOT/lib
#set Java_Home Environment Variables:
export JAVA_HOME=/opt/taobao/java
```

3. 执行 **source ~/.bashrc** 命令让环境变量配置生效。

2.4 安装动态库

安装动态库主要包括安装工具组、liblzo2、Snappy、libnuma、libaio、gtest、gmock 和其他动态库。如果您已经安装这些动态库，则可以跳过本章节。

2.4.1 安装工具组

编译 OceanBase 的脚本时，用到了 aclocal、autoconf 和 automake 等工具。因此我们需要安装 libtoolize（2.2.6 或以上版本），autoconf（2.66 或以上版本）和 automake（1.10.2 或以上版本）。

- 您可以执行 **libtoolize --version** 命令，查看 libtoolize 版本。
- 您可以执行 **autoconf --version** 命令，查看 autoconf 版本。
- 您可以执行 **automake --version** 命令，查看 automake 版本。

*** 安装 libtoolize**

安装 libtoolize 2.2.6 的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，下载“libtoolize”。
   wget http://mirrors.kernel.org/gnu/libtool/libtool-2.2.6b.tar.gz
3. 执行以下命令，解压缩安装包。
   tar zxf libtool-2.2.6b.tar.gz
4. 执行以下命令，进入安装目录。
   cd libtool-2.2.6b
5. 执行以下命令，安装 libtoolize。
   ./configure && make && sudo make install

*** 安装 autoconf**

安装 autoconf 2.66 的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，下载“autoconf”。
   wget http://ftp.gnu.org/gnu/autoconf/autoconf-2.66.tar.gz
3. 执行以下命令，解压缩安装包。
   tar zxf autoconf-2.66.tar.gz
4. 执行以下命令，进入安装目录。
   cd autoconf-2.66
5. 执行以下命令，安装 autoconf。
   ./configure && make && sudo make install
6. 执行以下命令，将“~/autoconf-2.66/bin/autoconf”文件拷贝到“/usr/bin”目录下。
   sudo \cp ~/autoconf-2.66/bin/autoconf /usr/bin

*** 安装 automake**

安装 automake 1.11.1 操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，下载“automake”。
   wget http://ftp.gnu.org/gnu/automake/automake-1.11.1.tar.gz
3. 执行以下命令，解压缩安装包。
   tar zxf automake-1.11.1.tar.gz
4. 执行以下命令，进入安装目录。
   cd automake-1.11.1
5. 执行以下命令，安装 automake。
   ./configure && make && sudo make install

6. 执行以下命令，将“~/automake-1.11.1/bin/automakef”文件拷贝到“/usr/bin”目录下。
   sudo \cp ~/automake-1.11.1/bin/automakef /usr/bin

2.4.2 安装 liblzo2

liblzo2 是一个压缩库，OceanBase 需要用它来压缩静态数据。

*** YUM 安装**

安装 liblzo2 的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，安装“lzo”。
   sudo yum install lzo

*** 手动安装**

如果 yum 无法找到 liblzo2 的包，则可以选择手动安装，操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，下载“liblzo2”的安装包。
   wget -c http://www.oberhumer.com/opensource/lzo/download/lzo-2.06.tar.gz
3. 执行以下命令，解压缩“lzo-2.06.tar.gz”。
   tar zxf lzo-*
4. 执行以下命令，进入“/home/admin/lzo-2.0.6”目录。
   cd lzo-2.06
5. 执行以下命令，编译并安装 liblzo2。
   ./configure --enable-shared --prefix=/usr/ && make && sudo make install

*** 验证**

安装完成后您可以编译一个 C 程序，验证 liblzo2 是否安装成功。

1. 在 OceanBase 服务器中输入以下代码：

```
echo "int main(){ return 0;}" > /tmp/a.c && gcc /tmp/a.c -llzo2 -o /tmp/a.out
```

2. 执行**/tmp/a.out** 命令，看是否报错。
   - 没有报错，则说明安装成功。
   - 显示以下的消息，则说明环境变量配置不正确。
     请将“liblzo2.so.2”的目录加入到“/home/admin/.bashrc”文件的“LD_LIBRARY_PATH”参数中。

```
./a.out: error while loading shared libraries: liblzo2.so.2: cannot open shared object file: No such file or directory
```

2.4.3 安装 Snappy

Snappy 是 Google 出品的压缩库。OceanBase 使用 Snappy 压缩静态数据。

注意：Snappy 依赖于 liblzo2，因此，安装 Snappy 前请先安装 liblzo2。

*** 安装**

安装 Snappy 的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，下载 Snappy 安装包。
   wget http://snappy.googlecode.com/files/snappy-1.0.3.tar.gz
3. 执行以下命令，解压缩“snappy-1.0.3.tar.gz”。
   tar -xvf snappy-1.0.3.tar.gz
4. 执行以下命令，进入 Snappy 的安装目录。
   cd snappy-1.0.3
5. 执行以下命令，安装 Snappy。
   ./configure && make -j 10 && sudo make install

*** 验证**

安装完成后你可以编译一个 C 程序，验证 Snappy 是否安装成功。

1. 在 OceanBase 服务器中输入以下代码：

   ```
   echo "int main(){ return 0;}" > /tmp/a.c && gcc /tmp/a.c -o /tmp/a.out -lsnappy
   ```

2. 执行**/tmp/a.out** 命令，看是否报错。
   - 没有报错，则说明安装成功。
   - 显示以下的消息，则说明环境变量配置不正确。
     请将“libsnappy.so.1”的目录加入到“/home/admin/.bashrc”文件的“LD_LIBRARY_PATH”参数中。

     ```
     ./a.out: error while loading shared libraries: libsnappy.so.1: cannot open shared object file: No such file or directory
     ```

2.4.4 安装 libnuma

Oceanbase 数据库中使用了 NUMA，因此需要 libnuma 支持。

安装 libnuma 的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，安装“libnuma”。
   sudo yum install numactl-devel.x86_64

2.4.5 安装 libaio

Oceanbase 中用到了 AIO，需要 libaio 的支持。下面通过安装 libaio 来添加 numa 相关的头文件和库。

*** YUM 安装**

安装 libaio 的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，安装“libaio”。
   sudo yum install libaio-devel.x86_64

*** 手动安装**

如果 yum 无法找到 libaio 的包，则可以选择手动安装，操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，下载 libaio 安装包。
   wget -c http://libaio.sourcearchive.com/downloads/0.3.107-7/libaio_0.3.107.orig.tar.gz
   说明：*如果该地址失效，请到"http://libaio.sourcearchive.com"手工下载。*
3. 执行以下命令，解压缩 libaio 安装包。
   tar zxf libaio*
4. 执行以下命令，进入 libaio 安装目录。
   cd libaio-0.3.107
5. 执行以下命令，编译安装 libaio。
   make && sudo make install

2.4.6 安装 gtest 和 gmock（可选）

如果您执行./configure --without-test-case 不编译 OceanBase 的 test，则不需要安装 gtest 和 gmock。

*** 安装 gtest**

安装 gtest 的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，下载 gtest 安装包。
   wget http://googletest.googlecode.com/files/gtest-1.6.0.zip
3. 执行以下命令，解压缩“gtest-1.6.0.zip”。
   unzip gtest-1.6.0.zip
4. 执行以下命令，进入 gtest 的安装目录。
   cd gtest-1.6.0

5. 依次执行以下命令，安装 gtest。
   ./configure && make
   sudo cp -r include/gtest /usr/local/include
   sudo cp -r lib/.libs/* /usr/local/lib/

*** 安装 gmock**

安装 gmock 的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，下载 gmock 安装包。
   wget http://googlemock.googlecode.com/files/gmock-1.6.0.zip
3. 执行以下命令，解压缩 gmock 安装包。
   unzip gmock-1.6.0.zip
4. 执行以下命令，进入 gmock 的安装目录。
   cd gmock-1.6.0
5. 依次执行以下命令，安装 gmock。
   ./configure && make
   sudo cp -r include/gmock /usr/local/include
   sudo cp -r lib/.libs/* /usr/local/lib/

2.4.7 安装其他库

在编译 OceanBase 时，还需要使用“openssl-devel”、“readline-devel”、“ncurses-devel 和“mysql-devel”四个库。

安装这些库的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 依次执行以下命令，安装 “openssl-devel”、“readline-devel”、“ncurses-devel”和“mysql-devel”。
   sudo yum install openssl-devel
   sudo yum install readline-devel
   sudo yum install ncurses-devel
   sudo yum install mysql-devel

2.5 安装 JDK（可选）

如果您需要采用 java 客户端访问 OceanBase 数据库，或者使用 OceanBase 的数据导入功能，则需要安装 JDK。

安装 JDK 的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，安装 JDK。
   sudo yum install java-1.6.0-openjdk-devel

2.6 创建数据磁盘挂载点

数据磁盘用于存放 Update Server 和 Chunk Server 的数据。如果您挂载磁盘，那么 Update Server 和 Chunk Server 的数据将存放到挂载的磁盘中，否则，将存放在挂载点中。

创建 Update Server 和 Chunk Server 数据磁盘挂载点的操作步骤如下：

1. 以 **admin** 登录 OceanBase 服务器。
2. 执行以下命令，创建磁盘挂载目录。
   sudo mkdir /data
3. 执行以下命令，将“/data”目录赋给“admin”用户。
   sudo chown admin /data
4. 根据磁盘规划和服务器规划创建挂载点。
   - Update Server
     for disk in {1..4}; do mkdir -p /data/$disk; done;
   - Chunk Server
     for disk in {1..10}; do mkdir -p /data/$disk; done;

3 安装 tbsys 和 tbnet

tbsys 主要对操作系统服务进行封装，tbnet 主要提供网络框架。OceanBase 依赖于这两个库。

*** 安装**

安装 tbsys 和 tbnet 的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，下载 tbsys 和 tbnet 的安装包。
   svn checkout http://code.taobao.org/svn//tb-common-utils/trunk/ tb-common-utils
3. 执行以下命令，进入 tbsys 和 tbnet 的安装目录。
   cd ~/tb-common-utils
4. 执行以下命令，编译安装 tbsys 和 tbnet。
   环境变量文件“/home/admin/.bashrc”中的“TBLIB_ROOT”参数所指示的目录下会生成“include”和“lib”两个子目录。
   sh build.sh

*** 验证**

安装成功后，可以采用如下方法验证编译器能否找到库：

1. 在 OceanBase 服务器中输入以下代码。

   ```
   echo "int main(){ return 0;}" > /tmp/a.c && gcc /tmp/a.c -o /tmp/a.out -L$TBLIB_ROOT/lib -ltbnet -ltbsys
   ```

2. 执行**/tmp/a.out** 命令，运行“a.out”。
   - 如果没报错，则说明安装成功。
   - 如果报错，请检查“/home/admin/.bashrc”文件中的“TBLIB_ROOT”参数是否配置正确。

4 安装 OceanBase

安装 OceanBase 的主要方式有两种：通过 RPM 包安装和通过源码安装。

如果您是普通用户建议您采用 RPM 安装；如果您是开发人员，建议您采用源码安装。

如果您需要采用 Root Server 主备双机，请先安装 HA，详细请参见《线上 HA 自动 BUILD 工具》。

4.1 采用 RPM 安装

如果您采用源码安装，则可以跳过本小节。

采用 RPM 安装 OceanBase 包括下载安装包、安装 libeasy、 安装 OceanBase 软件、配置免登录和配置一键启动及初始化。

4.1.1 下载安装包

下载 libeasy 和 OceanBase 安装包的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，下载 libeasy 和 OceanBase 安装包。
   git clone https://github.com/alibaba/oceanbase oceanbase_install
   下载的安装包说名如**表 4-1** 所示。

表 4-1 安装包说明

分支	安装包	说明	存放位置
oceanbase_0.4	-	OceanBase 0.4 的安装源码。	存放在分支的起始目录。
	t_libeasy-1.0.13-183.el5.x86_64.rpm	Linux 版本为 RedHat	存放在分支的“libeasy_rpm”文

分支	安装包	说明	存放位置
	t_libeasy-devel-1.0.13-183.el5.x86_64.rpm	5 的 libeasy 安装包。采用源码安装时需要安装。 ***说明：****您可以执行* ***cat /etc/issue*** *命令查看 Linux 版本号。*	件夹中。
	t_libeasy-1.0.13-183.el6.x86_64.rpm	Linux 版本为 RedHat 6 的 libeasy 安装包。采用源码安装时需要安装。	
	t_libeasy-devel-1.0.13-183.el6.x86_64.rpm		
	oceanbase-0.4.1.2-1105.el5.x86_64.rpm	Linux 版本为 RedHat 5 的 OceanBase 0.4 的 RPM 安装包。	存放在“oceanbase_rpm”文件夹中。
	oceanbase-0.4.1.2-1105.el6.x86_64.rpm	Linux 版本为 RedHat 6 的 OceanBase 0.4 的 RPM 安装包。	
oceanbase_0.3	-	OceanBase 0.3 的安装源码。	存放在分支的起始目录。

注：*“-”表示无。*

4.1.2 安装 OceanBase 软件

安装 OceanBase 软件操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，进入安装目录。
   cd ~/oceanbase_install
3. 执行以下命令，切换到 OceanBase 0.4 分支。
   git checkout oceanbase_0.4
4. 执行以下命令，进入“oceanbase_rpm”目录。
   cd ~/oceanbase_install/oceanbase_rpm
5. 执行以下命令，安装 OceanBase。
   sudo rpm --nodeps -ivh oceanbase-0.4.1.2-1105.el6.x86_64.rpm --prefix=/home/admin/oceanbase --force

4.1.3 配置免登录

在采用 RPM 安装时，需要在 OceanBase 的安装服务器中选择一台作为本机，配置该服务器到所有安装服务器的免登录。配置免登录后，该服务器在连接其他服务器时，无需输入密码。

本文档以单机部署为例，且选取该服务器作为本机，因此只需配置“本机到本机”的免登录。

*** 配置过程**

配置免登录的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，进入“.ssh”目录。
   cd ~/.ssh
   说明：*如果“.ssh”目录不存在，请先执行* ***mkdir ~/.ssh*** *命令创建。*
3. 执行以下命令，并按“Enter”键，直至生成公钥。
   ssh-keygen -t rsa
4. 执行以下命令，并根据提示输入登录密码，配置免登录。
   ssh-copy-id admin@10.10.10.2

*** 验证**

配置免登录完成后，您可以在任何一个目录下，输入“**ssh admin@10.10.10.2**”。

- 如果无需输入密码，则表示配置免登录成功。
- 如果仍需要输入密码，则请重新配置免登录。

4.1.4 配置一键启动及初始化

配置一键启动及初始化的方法如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，进入安装目录。
   cd ~/oceanbase_install
3. 执行以下命令，切换到 OceanBase 0.4 分支。
   git checkout oceanbase_0.4
4. 执行以下命令，进入一键启动脚本存放目录。
   cd ~/oceanbase_install/script/deploy
5. 执行以下命令，复制配置文件。
   cp oceanbase.conf.template deploy.conf
6. 使用 **vi** 编辑器，修改配置文件，如黑体部分所示。参数说明见注释部分。
   注意：*所有以"#@"开始的行有特殊含义，不允许当注释删除。*

```
#@begin_global  [settings]
# rs_admin 工具的位置，请勿修改。
rs_admin=./bin/rs_admin
# OceanBase 的安装目录。
ob_home=/home/admin/oceanbase
[public]
# APP 名称。
appname=obtest
# 主集群 ID，与集群名称对应，即以#@begin_cluster_x 和#@end_cluster_x 开头的行。
# OceanBase 内部使用纯数字 ID，即该配置中的数字部分为 ob 内部使用的集群 ID 号。
# 如果不指定集群 ID，则默认使用数字最小的集群为主集群。
master_cluster_id=cluster_1
# 网络接口名称，默认是 bond0。放到不同的 section 下可以单独为那个 section 中的 server 进行配置。
devname=bond0
[rootserver]
# Root Server 的服务端口。
port=3500
# Root Server 存放 commitlog 的目录。
# 执行脚本后，会在“/home/admin/oceanbase/data”下创建“rs_commitlog”目录，并软连接到“/data/log/rs_commitlog”。
commitlog_dir=/data/log/rs_commitlog
[chunkserver]
# Chunk Server 的端口。
port=2600
# Chunk Server 使用的磁盘数。
# 需要已经建立/data/{1..max_disk_num}的目录。
max_disk_num=10
[mergeserver]
# Merge Server 的服务端口。
port=2800
# Merge Server 的 MySQL 端口。
sql_port=3030
# 部署在 Root Server 上的 Listener 端口。
# 请勿修改！
```

```
lms_port=2828
[updateserver]
# Update Server 的服务端口。
port=2700
# Update Server 用于每日合并的端口。
inner_port=2710
# Update Server 转储用的磁盘的数目。
# 需要已经建立/data/{1..max_disk_num}的目录。
max_disk_num=4
#Update Server 存放 commitlog 的目录。
# 执行脚本后，会在“/home/admin/oceanbase/data”下创建“ups_commitlog”目录，并软连接到“/data/log/ups_commitlog”。
commitlog_dir=/data/log/ups_commitlog
#@end_global

#@begin_init_config
# 各 Server 启动时使用的配置项。
[rootserver]

[chunkserver]

[mergeserver]

[updateserver]
log_sync_type=1
#@end_init_config

#@begin_cluster_1
[public]
[rootserver]
# Root Server 的 vip 地址。vip 为 Root Server 的虚拟 IP。
# 主备 Root Server 时，获得 vip 的为主 Root Server。
vip=10.10.10.2
# 主备 Root Server 的 IP 地址。
10.10.10.2
[updateserver]
# 主备 Update Server 的 IP 地址。
10.10.10.2
[chunkserver]
# 所有 Chunk Server 的 IP 地址。
10.10.10.2
[mergeserver]
# 所有 Merge Server 的 IP 地址。
10.10.10.2
#@end_cluster_1

##@begin_cluster_2
## 多集群时需要配置，详细请参考 cluster_1。
##@end_cluster_2
```

7. 执行以下命令，一键启动及初始化。参数说明如表 4-2 所示，其他脚本命令请参见“7.1.4 一键脚本操作”。

   ./oceanbase.pl init --force -c 1 deploy.conf

表 4-2 参数说明

参数	说明
oceanbase.pl	运行脚本名称。
init	操作类型。初始化环境、启动并初始化集群。在首次安装时使用。
--force	强制执行，可省略。
-c 1	只对 cluster_1 进行初始化。如果不指定集群 ID，则初始化配置文件中的所有集群。
deploy.conf	配置文件名称。

4.2 采用源码安装

如果您采用 RPM 安装，则可以跳过本小节。

采用源码安装包括下载安装包、安装 libeasy、安装 OceanBase 软件、创建各 Server 所需目录、启动各 Server 服务和初始化 OceanBase。

4.2.1 下载安装包

下载 libeasy 和 OceanBase 安装包的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，下载 libeasy 和 OceanBase 安装包。
   git clone https://github.com/alibaba/oceanbase oceanbase_install
   下载的安装包说名如表 4-3 所示。

表 4-3 安装包说明

分支	安装包	说明	存放位置
oceanbase_0.4	-	OceanBase 0.4 的安装源码。	存放在分支的起始目录。
	t_libeasy-1.0.13-183.el5.x86_64.rpm	Linux 版本为 RedHat	存放在分支的“libeasy_rpm”文

分支	安装包	说明	存放位置
	t_libeasy-devel-1.0.13-183.el5.x86_64.rpm	5 的 libeasy 安装包。采用源码安装时需要安装。 ***说明：****您可以执行* ***cat /etc/issue*** *命令查看 Linux 版本号。*	件夹中。
	t_libeasy-1.0.13-183.el6.x86_64.rpm	Linux 版本为 RedHat 6 的 libeasy 安装包。采用源码安装时需要安装。	
	t_libeasy-devel-1.0.13-183.el6.x86_64.rpm		
	oceanbase-0.4.1.2-1105.el5.x86_64.rpm	Linux 版本为 RedHat 5 的 OceanBase 0.4 的 RPM 安装包。	存放在“oceanbase_rpm”文件夹中。
	oceanbase-0.4.1.2-1105.el6.x86_64.rpm	Linux 版本为 RedHat 6 的 OceanBase 0.4 的 RPM 安装包。	
oceanbase_0.3	-	OceanBase 0.3 的安装源码。	存放在分支的起始目录。

注：*“-”表示无。*

4.2.2 安装 libeasy

libeasy 是 Oceanbase 中新的网络通讯框架。

安装 libeasy 的操作步骤如下：

1. 执行以下命令，进入安装目录。
   cd ~/oceanbase_install
2. 执行以下命令，切换到 libeasy 分支。
   git checkout oceanbase_0.4
3. 执行以下命令，进入“libeasy_rpm”目录。
   cd ~/oceanbase_install/libeasy_rpm
4. 依次执行以下命令，安装 libeasy。
   sudo rpm -ivh t_libeasy-1.0.13-183.el6.x86_64.rpm --prefix=$EASY_LIB_PATH --force
   sudo rpm -ivh t_libeasy-devel-1.0.13-183.el6.x86_64.rpm --prefix=$EASY_ROOT --force
5. 用 **vi** 编辑器在“/home/admin/.bashrc”文件中，修改环境变量，如黑体部分所示。

```
export TBLIB_ROOT=~/tb-common-utils
export
LD_LIBRARY_PATH=/home/admin/oceanbase/lib:/usr/local/lib/libsnappy.so:/usr/:/usr/lib/:/usr/local/lib:/lib:$TBLIB_ROOT/lib
export EASY_ROOT=$TBLIB_ROOT
export EASY_LIB_PATH=$EASY_ROOT/lib/lib64
export JAVA_HOME=/opt/taobao/java
```

6. 执行以下命令，使坏境变量生效。
   source ~/.bashrc

4.2.3 安装 OceanBase 软件

安装 OceanBase 软件操作步骤如下：

1. 执行以下命令，进入安装目录。
   cd ~/oceanbase_install
2. 执行以下命令，切换到 OceanBase 0.4 的分支。
   git checkout oceanbase_0.4
3. 执行以下命令，初始化安装。
   sh build.sh init
4. 执行以下命令，指定安装目录“/home/admin/oceanbase”。
   ./configure --prefix=/home/admin/oceanbase --with-release=yes --with-test-case=no;
5. 依次执行以下命令，编译安装程序。
   make -j -C src/
   make -j -C tools/
6. 执行以下命令，安装 OceanBase。
   make install

7. 执行以下命令，进入“io_fault”目录。
   cd ~/oceanbase_install/tools/io_fault/
8. 执行以下命令，编译 tool 工具。
   make
9. 依次执行以下命令，将“libnone.so”文件拷贝到“~/oceanbase/lib/”目录下。
   cd /home/admin/oceanbase_install/src/common/compress/.libs
   \cp * ~/oceanbase/lib/

4.2.4 创建各 Server 所需目录

创建 Root Server、Update Server、Chunk Server 和 Merge Server 所需目录操作步骤如下：

1. 以 **admin** 登录 OceanBase 服务器。
2. 执行以下命令，创建 sstable 存放的目录。
   注意：*“~/obtest/sstable”目录在挂载到 Chunk Server 的磁盘中创建，并且“obtest”与 APP 名称相同。*
   for disk in {1..10}; do mkdir -p /data/$disk/obtest/sstable; done;
3. 执行以下命令，创建数据存放目录。
   mkdir -p /home/admin/oceanbase/data
4. 执行以下命令，创建 Root Server 和 Update Server 所需目录。
   - Root Server
     mkdir -p /home/admin/oceanbase/data/rs
     mkdir -p /home/admin/oceanbase/data/rs_commitlog
   - Update Server
     mkdir -p /home/admin/oceanbase/data/ups_commitlog
     mkdir -p /home/admin/oceanbase/data/ups_data/raid0
     mkdir -p /home/admin/oceanbase/data/ups_data/raid1
5. 执行以下命令，建立 Chunk Server 和 Update Server 与数据存放磁盘的软连接。
   - Chunk Server
     for disk in {1..10}; do ln -s /data/$disk
     /home/admin/oceanbase/data/$disk; done;
   - Update Server
     ln -s /data/1
     /home/admin/oceanbase/data/ups_data/raid0/store0
     ln -s /data/2
     /home/admin/oceanbase/data/ups_data/raid0/store1
     ln -s /data/3
     /home/admin/oceanbase/data/ups_data/raid1/store0
     ln -s /data/4
     /home/admin/oceanbase/data/ups_data/raid1/store1

4.2.5 启动各 Server 服务

启动 Root Server、Update Server、Chunk Server 和 Merge Server 的方法如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，进入 OceanBase 安装目录。
   cd /home/admin/oceanbase
3. 执行以下命令，启动 Root Server、Update Server、Chunk Server 和 Merge Server。参数说明如表 4-4 所示。
   注意：*启动 Chunk Server 前请先启动 Root Server，否则 Chunk Server 在一段时间后会自动结束进程。*
   - 启动 Root Server
     bin/rootserver -r 10.10.10.2:3500 -R 10.10.10.2:3500 -C 1
   - 启动 Update Server
     bin/updateserver -r 10.10.10.2:3500 -p 2700 -m 2710
   - 启动 Chunk Server
     bin/chunkserver -r 10.10.10.3:3500 -p 2600 -n obtest
     说明：*obtest 为“4.2.4 创建各 Server 所需目录”中创建 sstable 所在的目录。*
   - 启动 Merge Server
     bin/mergeserver -r 10.10.10.3:3500 -p 2800 -z 3030

表 4-4 参数解释

服务器	参数	说明
Root Server	-r	当前 Root Server 的 IP 地址和并设置服务端口。 格式：-r [IP]:[Port]
	-R	主集群中的主 Root Server 的 IP 地址和端口。 格式：-R [IP]:[Port]
	-C	设置集群 ID，必须为数字。 格式：-C [Cluster ID]
Update Server	-r	需要链接的 Root Server 的 IP 地址和端口。 格式：-r [IP]:[Port]
	-p	设置当前 Update Server 的服务端口。 格式：-p [Port]

服务器	参数	说明
	-m	每日合并操作时，Chunk Server 请求合并数据所用的端口。 格式：-m [Port]
Chunk Server	-r	需要链接的 Root Server 的 IP 地址和端口。 格式：-r [IP]:[Port]
	-p	设置当前 Chunk Server 的服务端口。 格式：-p [Port]
	-n	APP 名称。与“3.9 创建各 Server 所需目录”中 sstable 的父目录名称保持一致。 格式：-n [APP Name]
Merge Server	-r	需要链接的 Root Server 的 IP 地址和端口。 格式：-r [IP]:[Port]
	-p	设置当前 Merge Server 的服务端口。 格式：-p [Port]
	-z	设置 MySQL 的协议端口。 格式：-z [Port]

4.2.6 初始化 OceanBase

初始化 OceanBase 的操作步骤如下：

1. 以 **admin** 用户登录 OceanBase 服务器。
2. 执行以下命令，进入“/home/admin/oceanbase/bin”目录。
   cd /home/admin/oceanbase/bin
3. 依次执行以下命令，初始化 OceanBase，参数说如表 4-5 所示。
   ./rs_admin -r 10.10.10.2 -p 3500 set_obi_role -o OBI_MASTER
   ./rs_admin -r 10.10.10.2 -p 3500 -t 60000000 boot_strap

   表 4-5 参数说明

参数	说明
-r	Root Server 的 IP 地址 格式：-r [IP]

参数	说明
-p	Root Server 的端口号。 格式：-r [Port]
-o	指定主集群 Root Server。 格式：set_obi_role -o OBI_MASTER ***注意：****如果您安装 OceanBase 集群，还需要指定备集群的 Root Server，格式为：set_obi_role -o OBI_SLAVE*
-t	命令的超时时长。 单位：微秒。 格式：-t [Time] boot_strap

小窍门：*在“/home/admin/oceanbase/bin”目录下，执行****./rs_admin*** *命令，可以查看 help 信息。*

配置成功后，系统显示如下：

```
[admin@obtest-1-2  ztt.alipay.net /home/admin/oceanbase/bin]
$ ./rs_admin -r 10.10.10.2 -p 3500 set_obi_role -o OBI_MASTER
timeout=10000000
set_obi_role...role=0
Okay

[admin@obtest-1-2  ztt.alipay.net /home/admin/oceanbase/bin]
$ ./rs_admin -r 10.10.10.2 -p 3500 -t 60000000 boot_strap
timeout=60000000
do_rs_admin, cmd=16...
Okay
```

5 安装 MySQL 客户端

您需要在本地计算机中安装 MySQL 客户端链接 OceanBase。

*** 安装**

假设本地计算机的用户为 **sqluser**。安装客户端的操作步骤如下：

1. 以 **sqluser** 用户登录本地计算机。
2. 执行以下命令，安装 MySQL 客户端。
   sudo yum install mysql

*** 后续操作**

- 执行 **mysql -h 10.10.10.2 -P3030 -uadmin -padmin** 命令，链接 OceanBase。
  - IP 为 Merge Server 的 IP 地址。
  - 端口号为 MySQL 协议端口。
  - OceanBase 的初始“用户名/密码”为“admin/admin”。
- 执行 **exit** 命令，退出 OceanBase。
- 如果您想要详细了解 OceanBase 的使用，请参考《OceanBase SQL 参考指南》。

6 FAQ

6.1 启动 Update Server 时报错

*** 现象描述**

执行 **bin/updateserver -r 10.10.10.2:3500 -p 2700 -m 2710** 命令启动 Update Server 时，出现如下报错：

```
[2013-04-12 09:07:01.513130] INFO ob_server_config.cpp:139 [140607873587520]
==================== *stop server config report* ======================
libnone.so: cannot open shared object file: No such file or directory
[2013-04-12 09:07:01.513823] ERROR check_all (ob_update_server_config.cpp:94)
[140607873587520] cannot load compressor library name=[none]
[2013-04-12 09:07:01.513849] WARN do_work (ob_update_server_main.cpp:98)
[140607873587520] failed to load from conf, ret: [1]
[2013-04-12 09:07:01.513939] ERROR do_work (ob_update_server_main.cpp:103)
[140607873587520] Start Update server failed, ret: [1]
[2013-04-12 09:07:01.513959] INFO base_main.cpp:405 [140607873587520] exit program.
```

*** 可能原因**

- OceanBase 自带的压缩库 libnone 环境变量配置错误。
- 当前用户对 log 和 run 文件没有写权限。

*** 处理方法**

- 将 libnone 路径添加到环境变量中，详细操作步骤如下：
  1. 以 **admin** 用户登录 OceanBase 服务器。
  2. 用 **vi** 编辑器在“/home/admin/.bashrc”文件中，添加 libnone 的安装路径，如黑体部分所示：

     ```
     export TBLIB_ROOT = ~/tb-common-utils
     export LD_LIBRARY_PATH =
     /home/admin/oceanbase/lib:/usr/local/lib/libsnappy.so:/usr/:/usr/lib/:/usr/loc
     al/lib:/lib:$TBLIB_ROOT/lib
     export EASY_ROOT = $TBLIB_ROOT
     export EASY_LIB_PATH = $EASY_ROOT/lib/lib64
     export JAVA_HOME = /opt/taobao/java
     ```

  3. 执行 **source ~/.bashrc** 命令让环境变量配置生效。
- 将“log”和“run”目录的拥有者修改为 admin，操作步骤如下：
  1. 以 **admin** 用户登录 OceanBase 服务器。

2. 执行以下命令，查看并记录当前用户所在组。
   groups
3. 执行以下命令，并输入密码，切换到 root 用户。
   su - root
4. 执行以下命令，进入“/home/obuser/oceanbase”目录。
   cd /home/obuser/oceanbase
5. 依次执行以下命令，修改“log”和“run”目录的拥有者为 admin。
   chown -R admin:admin log
   chown -R admin:admin run

6.2 安装 gcc 时编译出错

*** 现象描述**

安装 gcc 4.1.2 时，编译报错：

```
/usr/include/gnu/stubs.h:7:27: 错误：gnu/stubs-32.h：没有那个文件或目录
```

*** 可能原因**

glibc-devel 没有安装。

*** 处理方法**

安装 glibc-devel，详细操作步骤如下：

1. 以 root 用户登录 OceanBase 服务器。
2. 执行以下命令，安装 glibc-devel。
   - Ubuntu 操作系统
     sudo apt-get install libc6-dev-i386
   - Red Hat 操作系统
     yum install glibc-devel.i686
   - CentOS 5.8 操作系统
     yum install glibc-devel.i386
   - CentOS 6.3 操作系统
     yum install glibc-devel.i686
   - SLES 操作系统
     zypper in glibc-devel-32bit

7 附录

7.1 常用操作

7.1.1 启动服务

Root Server、Update Server、Chunk Server 和 Merge Server 的服务端口将在启动时设置。

启动 Root Server、Update Server、Chunk Server 和 Merge Server 服务方法如下：

1. 以 admin 用户登录 OceanBase 服务器。
2. 执行以下命令，进入 OceanBase 的安装目录。
   cd /home/admin/oceanbase
3. 执行以下命令，启动 Root Server、Update Server、Chunk Server 和 Merge Server。参数说明如表 7-1 所示。
   注意：*启动 Chunk Server 前请先启动 Root Server，否则 Chunk Server 在一段时间后会自动结束进程。*
   - 启动 Root Server
     bin/rootserver -r 10.10.10.2:3500 -R 10.10.10.2:3500 -C 1
   - 启动 Update Server
     bin/updateserver -r 10.10.10.2:3500 -p 2700 -m 2710
   - 启动 Chunk Server
     bin/chunkserver -r 10.10.10.2:3500 -p 2600 -n obtest
   - 启动 Merge Server
     bin/mergeserver -r 10.10.10.2:3500 -p 2800 -z 3030

表 7-1 参数解释

服务器	参数	说明
Root Server	-r	当前 Root Server 的 IP 地址和并设置服务端口。 格式：-r [IP]:[Port]

服务器	参数	说明
	-R	主集群中的主 Root Server 的 IP 地址和端口。 格式：-R [IP]:[Port]
	-C	设置集群 ID，必须为数字。 格式：-C [Cluster ID]
Update Server	-r	需要链接的 Root Server 的 IP 地址和端口。 格式：-r [IP]:[Port]
	-p	设置当前 Update Server 的服务端口。 格式：-p [Port]
	-m	每日合并操作时，Chunk Server 请求合并数据所用的端口。 格式：-m [Port]
Chunk Server	-r	需要链接的 Root Server 的 IP 地址和端口。 格式：-r [IP]:[Port]
	-p	设置当前 Chunk Server 的服务端口。 格式：-p [Port]
	-n	APP 名称。与“3.9 创建各 Server 所需目录”中 sstable 的父目录名称保持一致。 格式：-n [APP Name]
Merge Server	-r	需要链接的 Root Server 的 IP 地址和端口。 格式：-r [IP]:[Port]
	-p	设置当前 Merge Server 的服务端口。 格式：-p [Port]

服务器	参数	说明
	-z	设置 MySQL 的协议端口。 格式：-z [Port]

7.1.2 停止服务

在 OceanBase 服务器中，停止 Root Server、Update Server、Chunk Server 和 Merge Server 服务方法如下：

1. 以 obuser 用户登录 OceanBase 服务器。
2. 执行以下命令，停止 Root Server、Update Server、Chunk Server 和 Merge Server。

   注意：*停止各个服务时，不建议使用“kill -9”。*

   - 停止 Root Server
     killall rootserver
   - 停止 Update Server
     killall updateserver
   - 停止 Chunk Server
     killall chunkserver
   - 停止 Merge Server
     killall mergeserver

7.1.3 重新启动

如果您进行 OceanBase 数据库各 Server 的重新启动操作，请您遵守以下规则：

- 重新启动前，确保各个 Server 的进程已退出。
- 重新启动 Root Server 时，Cluster ID 与之前保持一致。同时必须重新指定主备。
- 重新启动 Chunk Server 时，App Name 与之前保持一致。
- 如果 OceanBase 为单机部署，启动不同 Server 的进程时，建议间隔 10 秒。

7.1.4 一键脚本操作

一键脚本的命令以及功能如下所示，参数说明如表 7-2所示。

- 初始化环境、启动并初始化集群。在首次安装时使用。
  ./oceanbase.pl init *[--force] [-c 1] deploy.conf*
- 启动服务，不初始化环境。
  ./oceanbase.pl start [--force] *deploy.conf*

- 停止服务。
  ./oceanbase.pl stop [--force] ***deploy.conf***
- 清除服务。系统将被还原到“init”前，请谨慎使用。
  ./oceanbase.pl clean [--force] ***deploy.conf***

  表 7-2 参数说明

参数	说明
oceanbase.pl	运行脚本名称。
init/start/stop/clean	操作类型。
--force	强制执行，可省略。
-c	只对 cluster_1 进行初始化。如果不指定集群 ID，则初始化配置文件中的所有集群。
deploy.conf	配置文件名称。

7.1.5 卸载

卸载 OceanBase 数据库只需要删除 OceanBase 的安装用户及目录即可，删除安装用户的操作步骤如下：

1. 以 root 用户登录 OceanBase 服务器。
2. 执行以下命令，停止 admin 下的所有进程。
   ps -ef |grep admin|awk '{print $2}' | xargs kill
3. 执行如下命令，删除 admin 用户及用户目录。
   userdel -r admin
4. 执行如下命令，删除数据文件。
   rm -rf /data
5. 执行如下命令，删除临时文件。
   rm -rf /tmp/*

7.2 安装 gcc 4.1.2

安装 gcc 4.1.2 的操作步骤如下：

1. 以 root 用户登录 OceanBase 服务器。
2. 执行以下命令，查看是否安装“makeinfo”。
   makeinfo --version
   - 已安装，则记录版本号，然后执行“步骤 3”。
   - 未安装，则执行 **yum install texinfo** 命令，安装“makeinfo”。

3. 执行以下命令，下载“gcc-4.1.2.tar.bz2”。
   wget ftp://ftp.gnu.org/gnu/gcc/gcc-4.1.2/gcc-4.1.2.tar.bz2
4. 执行以下命令，解压缩“gcc-4.1.2.tar.bz2”。
   tar -xvf gcc-4.1.2.tar.bz2
5. 执行以下命令，进入“gcc-4.1.2”目录。
   cd gcc-4.1.2
6. 使用 **vi** 编辑器，修改“configure”文件。如果您的“makeinfo”的版本在“4.2-4.9”之间，则跳过此步骤。
7. # For an installed makeinfo, we require it to be from texinfo 4.2 or
8. # higher, else we use the “missing” dummy.
9. if ${MAKEINFO} –version \

   | egrep **'texinfo[^0-9]*([1-3][0-9]|4\.[2-9]|[5-9])'** >/dev/null 2>&1;

   - “makeinfo”的版本为“4.13”，则将粗体部分修改为以下内容：

     ```
     'texinfo[^0-9]*([1-3][0-9]|4\.[4-9]|4\.[1-9][0-9]*|[5-9])'
     ```

   - “makeinfo”为其他版本，则将粗体部分修改为以下内容：

     ```
     'texinfo[^0-9]*([1-3][0-9]|4\.[2-9]|4\.[1-9][0-9]*|[5-9])'
     ```

   小窍门：*您可以在* ***vi*** *里使用****/texinfo[^0-9]****快速定位上面两行。*
10. 执行以下命令编译 gcc 4.1.2。
    ./configure --prefix=/usr/local/gcc-4.1.2&& make
11. 执行以下命令安装 gcc 4.1.2。
    make install
12. 执行以下命令，进入“/usr/bin”目录。
    cd /usr/bin
13. 依次执行以下命令，删除原有的 gcc 链接文件。
    rm –rf gcc
    rm –rf g++
14. 依次执行以下命令，建立 gcc 4.1.2 的链接。
    ln -s /usr/local/gcc-4.1.2/bin/gcc /usr/bin/gcc
    ln -s /usr/local/gcc-4.1.2/bin/g++ /usr/bin/g++
15. 执行以下命令，查看 gcc 版本。
    gcc –v

7.3 内部表参数说明

为了区别用户定义的表，OceanBase 的内部表的名称都以下划线“__”开头。

7.3.1 __first_tablet_entry

“__first_tablet_entry”记录了集群中所有 table 的基本属性信息。

Rowkey：(table_name)

“__first_tablet_entry”参数说明如表 7-3 所示。

表 7-3 __first_tablet_entry 参数

参数	类型	说明
gm_create	createtime	创建时间。
gm_modify	modifytime	修改时间。
table_name	varchar	表名。
creat_time_column_id	int	create_time 列的列 id。
modify_time_column_id	int	modify_time 列的列 id。
table_id	int	表 ID。
table_type	int	• 1：普通表。 • 2：索引。 • 3：元数据表。 • 4：view。 • 5：临时表。
load_type	int	• 1：保存到磁盘。 • 2：保存到内存。
table_def_type	int	• 1：内部表。 • 2：用户定义表。
rowkey_column_num	int	主键的列数，后续 endrowkeyobj1, endrowkeyobj2…等来依次表示主键的列。
column_num	int	全部的列数(包括主键)。
max_used_column_id	int	该表使用过的最大列 ID(列 ID 不重用)。
replica_num	int	单个集群的 tablet 的 replica 的个数(1~6)。

参数	类型	说明
create_mem_version	int	新建该表时候系统的 mem_version，暂时保留。
tablet_max_size	int	该表每个 tablet 的 sstable 文件最大允许大小。
max_rowkey_length	int	Rowkey 的最大长度限制。
compress_func_name	varchar	存储 sstable 所使用的压缩方法名称。
is_use_bloomfilter	int	指定是否使用 bloomfilter。
merge_write_sstable_version	int	合并的时候写哪个版本的 sstable
is_pure_update_table	int	指定是否属于内存更新表。
expire_condition	varchar	使用表达式定义的此表的数据自动过期删除条件。
rowkey_split	int	用于指定每日合并中 tablet 的分裂点为 rowkey 的第几个 obj。
tablet_block_size	int	Tablet_block 的大小。
is_read_static	int	是否要读静态数据。

7.3.2 __all_all_column

“__all_all_column”存储了每个表的所有列、column_id、列类型等，包括内部表(不包括核心表)和用户定义表。与内部表“__all_join_info”共同定义了各个表的 schema 信息。

Rowkey：(table_id，column_name)

“__all_all_column”参数说明如表 7-4 所示。

表 7-4 __all_all_column 参数

参数	类型	说明
gm_create	createtime	创建时间。
gm_modify	modifytime	修改时间。

参数	类型	说明
table_id	int	表 ID。
column_name	varchar	列名。
table_name	varchar	表名。
column_id	int	列 ID。
column_group_id	int	列隶属的 column group id。
rowkey_id	int	• 0：非 rowkey。 • 正整数：rowkey 的序号，必须是从 1 开始的连续正整。 ***说明：***“*__all_table_table*”中的“*rowkey_column_num*”定义了该表的 *rowkey* 的列数量。
length_in_rowkey	int	如果是 rowkey 列，表示在二进制 rowkey 串中占用的字节数。
order_in_rowkey	int	表示该列的升降序。
join_table_id	int	• -1：没有 join。 • 正整数：连接表的表 ID。
join_column_id	int	• -1：没有 join。 • 正整数：连接表中的列 ID。
data_type	int	数据类型。
data_length	int	整数的字节数或字符串的最大长度。
data_precision	int	整数的十进制位数或 decimal 的有效位数(小数点前和小数点后)。
data_scale	int	decimal 小数点后的位数。
nullable	int	• 1：不可以为空。 • 2：可以为空。

7.3.3 __all_join_info

“__all_join_info”存储了表之间的内部 join 关系，即左表通过其某些列对应到右表的 rowkey。

说明：OceanBase 只支持左连接，左表及右表的对应列的类型必须一致。

Rowkey：(left_table_id, left_column_id, right_table_id, right_column_id)

“__all_join_info”参数说明如表 7-5 所示。

表 7-5 __all_join_info 参数

参数	类型	说明
gm_create	createtime	创建时间。
gm_modify	modifytime	修改时间。
left_table_id	int	左表的表 ID。
left_column_id	int	左表的列 ID。
right_table_id	int	右表的表 ID。
right_column_id	int	右表的列 ID。
left_table_name	varchar	左表的表名。
left_column_name	varchar	左表的列名。
right_table_name	varchar	右表的表名。
right_column_name	varchar	右表的列名。

7.3.4 __all_client

“__all_client”用来保存 JAVA 客户端的版本信息。

Rowkey：(client_ip, version)

“__all_client”参数说明如表 7-6 所示。

表 7-6 __all_client 参数

参数	类型	说明
gm_create	createtime	创建时间。

参数	类型	说明
gm_modify	modifytime	修改时间。
client_ip	varchar	• 0：与特定集群无关。 • 正整数：指定集群。
version	varchar	客户端版本。
status	varchar	客户端状态。
extra1	varchar	预留。
extra2	int	预留。

7.3.5 __all_cluster

“__all_cluster”记录了系统中所有的集群，这个表由每个集群的主 RootServer 更新。

Rowkey：(cluster_id)

“__all_cluster”参数说明如表 7-7 所示。

表 7-7 __all_cluster 参数

参数	类型	说明
gm_create	createtime	创建时间。
gm_modify	modifytime	修改时间。
cluster_id	int	集群 ID，正整数。
cluster_vip	varchar	Cluster 的 IP 地址。
cluster_port	int	Cluster 端口号。
cluster_role	int	• 1： slave。 • 2：master。
cluster_name	varchar	集群名称。
cluster_info	varchar	集群说明信息。

参数	类型	说明
cluster_flow_percent	int	流量配比。
read_strategy	int	客户端使用的负载均衡策略： • 0：随机轮转策略。 • 1：一致性哈希。

7.3.6 __all_server

“__all_server”记录了系统中所有的服务器，这个表仅仅由主集群的主 RootServer 更新。

Rowkey：(cluster_id，svr_type, svr_ip, svr_port)

“__all_server”参数说明如表 7-8 所示。

表 7-8 __all_server 参数

参数	类型	说明
gm_create	createtime	创建时间。
gm_modify	modifytime	修改时间。
cluster_id	int	• 0：与特定集群无关。 • 正整数：指定的集群 ID。
svr_type	varchar	• RootServer • ChunkServer • MergeServer • UpdateServer • Other
svr_ip	varchar	Server IP 地址。
svr_port	int	Server 端口。
inner_port	int	内部交互端口。

参数	类型	说明
svr_role	int	• 0：与服务器角色(RS 或 UPS 的主或备)无关。 • 1：slave • 2：master
svr_version	varchar	程序版本信息。

7.3.7 __all_server_stat

“__all_server_stat”用于记录本集群内所有服务器的监控信息，属于虚拟的内存表，不对监控数据进行存储。

Rowkey：(svr_type, svt_ip, svr_port, name)

“__all_server_stat”参数说明如表 7-9 所示。

表 7-9 __all_server_stat 参数

参数	类型	说明
svr_type	varchar	• RootServer • ChunkServer • MergeServer • UpdateServer • Other
svr_ip	varchar	Server IP 地址。
svr_port	int	Server 端口。
name	varchar	监控项的名称。
value	int	监控项的值。

7.3.8 __all_sys_config

“__all_sys_config”存储了 Server 所需的配置项参数。

Rowkey：(cluster_id，svr_type，svr_ip, svr_ port, name)

“__all_sys_config”参数说明如表 7-10 所示。

表 7-10 __all_sys_config 参数

参数	类型	说明
gm_create	createtime	创建时间。
gm_modify	modifytime	修改时间。
cluster_id	int	• 0：与特定集群无关。 • 正整数：指定集群。
svr_type	varchar	• RootServer • ChunkServer • MergeServer • UpdateServer • Other
svr_ip	varchar	Server IP 地址。
svr_port	int	Server 端口。
name	varchar	参数名称。
section	varchar	参数所属的段。
data_type	varchar	参数值的数据类型。
value	varchar	参数值。
value_strict	varchar	参数值的约束。
info	varchar	对该项的说明。

7.3.9 __all_sys_config_stat

“__all_sys_config_stat”用于显示当前各个 Server 已经生效的配置项参数值，它的 Schema 与“__all_sys_config”的相同。

Rowkey：(cluster_id，svr_type，svr_ip, svr_ port, name)

“__all_sys_config_stat”参数说明如表 7-11 所示。

表 7-11 __all_sys_config_stat 参数

参数	类型	说明
gm_create	createtime	创建时间。
gm_modify	modifytime	修改时间。
cluster_id	int	• 0：与特定集群无关。 • 正整数：指定集群。
svr_type	varchar	• RootServer • ChunkServer • MergeServer • UpdateServer • Other
svr_ip	varchar	Server IP 地址。
svr_port	int	Server 端口。
name	varchar	参数名称。
section	varchar	参数所属的段。
data_type	varchar	参数值的数据类型。
value	varchar	参数值。
value_strict	varchar	参数值的约束。
info	varchar	对该项的说明。

7.3.10 __all_sys_param

“__all_sys_param”存储了系统所需的诸多参数，如环境变量等，不同的参数保存在不同行。

Rowkey：(cluster_id,name)

“__all_sys_param”参数说明如表 7-12 所示。

表 7-12 __all_sys_param 参数

参数	类型	说明
gm_create	createtime	创建时间。
gm_modify	modifytime	修改时间。
cluster_id	int	• 0：与特定集群无关。 • 正整数：指定的集群 ID。
name	varchar	参数名称。
data_type	int	参数值的数据类型。
value	varchar	参数值。
info	varchar	对该项的说明。

在“__all_sys_param”表中已定义的参数说明如表 7-13 所示。

表 7-13 已定义的参数

参数		说明
name	**data_type**	
autocommit	int	是否自动提交。
character_set_results	vchar	字符集。
max_allowed_packe	int	最大网络包大小。
ob_app_name	vchar	应用的名称。
ob_group_agg_push_down_param	bool	聚合操作是否下移到 Chunkserver 的开关。
ob_tx_idle_timeout	int	事务开始后无任何操作时，事务的超时时间。
ob_read_consistency	int	读一致性级别。
ob_tx_timeout	int	事务超时时间。
tx_isolation	vchar	事务隔离性。

参数		说明
sql_mode	vchar	SQL 模式。

7.3.11 __all_sys_stat

“__all_sys_stat”存储了系统各种状态值，不同的项保存在不同行。

Rowkey：(cluster_id，name)

“__all_sys_stat”参数说明如表 7-14 所示。

表 7-14 __all_sys_stat 参数

参数	类型	说明
gm_create	createtime	创建时间。
gm_modify	modifytime	修改时间。
cluster_id	int	• 0：与特定集群无关。 • 正整数：指定集群。
name	varchar	参数名称。
data_type	int	值的类型。
value	varchar	参数值。
info	varchar	对参数的说明。

在“__all_sys_stat”表中已经定义的参数说明如表 7-15 所示。

表 7-15 已定义的参数

参数			说明
name	**cluster_id**	**data_type**	
max_used_table_id	0	int	已经使用的最大 table_id。
max_used_user_id	0	int	已经使用的最大 user_id。

7.3.12 __all_table_privilege

“__all_table_privilege”记录了系统中用户在每个表的读写等权限。

Rowkey：(user_id，table_id)

“__all_table_privilege”参数说明如表 7-16 所示

表 7-16 __all_table_privilege 参数

参数	类型	说明
gm_create	createtime	创建时间。
gm_modify	modifytime	修改时间。
user_id	int	用户内部 ID。
table_id	int	表 ID，table_id = 0 时，表示 all_table。
priv_all	int	是否有所有权限。
priv_alter	int	是否有 alter table 权限。
priv_create	int	是否有 create table 权限。
priv_create_user	int	是否有 create user 权限。
priv_delete	int	是否有 delete table 权限。
priv_drop	int	是否有 drop table 权限。
priv_grant_option	int	是否有 grant 授权权限。
priv_insert	int	是否有 insert 权限。
priv_update	int	是否有 update 权限。
priv_select	int	是否有 select 权限。
priv_replace	int	是否有 replace 权限。

7.3.13 __all_trigger_event

“__all_trigger_event”用于记录内部通知事件。

Rowkey：(event_ts)

“__all_trigger_event”参数说明如表 7-17 所示。

表 7-17 __all_trigger_event 参数

参数	类型	说明
gm_create	createtime	创建时间。
gm_modify	modifytime	修改时间。
event_ts	PrecisDateTime	事件发生时间戳。
src_ip	varchar	事件发生源机器 ip。
event_type	int	事件类型。
event_param	int	消息参数。
extra	varchar	预留。

7.3.14 __all_user

“__all_user”记录了系统中所有的可以登录 OceanBase 的用户，每个用户一行记录。

Rowkey：(user_name)

“__all_user”参数说如表 7-18 所示。

表 7-18 __all_user 参数

参数	类型	说明
gm_create	createtime	创建时间。
gm_modify	modifytime	修改时间。
user_name	varchar	用户名。
user_id	int	用户内部 ID。
pass_word	varchar	用户密码（密文存储）。
info	varchar	注释。
priv_all	int	是否拥有所有的权限。
priv_alter	int	是否有 alter 权限。

参数	类型	说明
priv_create	int	是否有 create table 权限。
priv_create_user	int	是否有 create user 权限。
priv_delete	int	是否有 delete table 权限。
priv_drop	int	是否有 drop table 权限。
priv_grant_option	int	是否有 grant 授权权限。
priv_insert	int	是否有 insert 权限。
priv_update	int	是否有 update 权限。
priv_select	int	是否有 select 权限。
priv_replace	int	是否有 replace 权限。
is_locked	int	是否被锁。

7.4 配置配置参数说明

本章节介绍 OceanBase0.4 的所有配置参数。查看、修改系统参数的方法请参见《OceanBase SQL 参考手册》的“5.4 修改系统配置项”章节。

7.4.1 Root Server 配置参数

Root Server 配置参数说明如表 7-19 所示。

表 7-19 Root Server 配置参数

参数	缺省值	说明
balance_max_concurrent_migrate_num	2	Max concurrent migrate num for balance. Range: [1,10]
balance_max_migrate_out_per_cs	20	Max migrate out per Chuck Server. Range: [1,100]

参数	缺省值	说明
balance_max_timeout	5m	Max timeout time for one group of balance task. You are advised to use the default value.
balance_timeout_delta	10s	Balance timeout delta. You are advised to use the default value.
balance_tolerance_count	10	Tolerance count for balance. Range: [1,1000]
balance_worker_idl_time	30s	Balance worker idle wait time. You are advised to use the default value.
cluster_id	-	Cluster id.
commit_log_dir	data/rs_commitlog	Root Server commit log directory.
commit_log_sync_type	1	Commit log sync type.
create_table_in_init	False	True: Create table while init process. False: Not create tablet while init process.

参数	缺省值	说明
cs_lease_duration_time	10s	Chunk Server lease duration time. You are advised to use the default value.
cs_probation_period	5s	Duration before chunkserver can adopt migrate.
devname	bound0	Listen device.
enable_balance	True	True: Balance switch on. False: Balance switch off.
enable_cache_schema	True	True: Cache schema in RootServer. False: Not to cache schema in RootServer.
enable_new_root_table	False	True: Use new root_table data structure False: No use new root_table data structure
enable_rereplication	True	True: Rereplication switch on. False: Rereplication switch off.
expected_request_process_time	10ms	Expected request process time, check before pushing in task queue.

参数	缺省值	说明
first_meta_filename	first_tablet_meta	First meta file name.
io_thread_count	4	I/O thread count. Must be restarted to take effect after modifying the parameter. Range: [1,100]
is_import	False	True: Import application. False: Not import application.
lease_interval_time	15s	Lease interval time. You are advised to use the default value.
lease_on	True	True: Lease between master and slave rootserver. False: No Lease between master and slave rootserver.
lease_reserved_time	10s	Lease reserved time. You are advised to use the default value.
log_queue_size	100	Log queue size. Range: [10,100000]

参数	缺省值	说明
log_replay_wait_time	100ms	Log replay wait time. You are advised to use the default value.
log_sync_limit	40MB	Log synchronization limit.
log_sync_timeout	500ms	Log synchronization timeout time. You are advised to use the default value.
master_root_server_ip	0.0.0.0	Master OceanBase instance Virtual IP address.
master_roo_server_port	0	Master OceanBase instance listen port.
max_commit_log_size	64MB	Max commit log size. You are advised to use the default value.
max_merge_duration_time	2h	Max merge duration time.
migrate_wait_time	60s	Waiting time for balance thread to start work. You are advised to use the default value.

参数	缺省值	说明
network_timeout	50s	Network timeout for remote procedure call.
obconnector_port	5433	Obconnector port.
port	3500	Root server listen port. Range: (1024,65535)
read_master_master_ups_percent	100	Master master Update Server read percent. Range: [0,100]
read_queue_size	500	Read queue size. Range: [10,100000]
read_slave_master_ups_percent	100	Slave master Update Server read percent. Range: [0,100]
read_thread_count	20	Read thread count. Must be restarted to take effect after modifying the parameter. Range: [10,100]
retry_times	3	Retry times if failed.
root_server_ip	-	Root server' IP.
rs_data_dir	data/rs	Root Server data directory.

参数	缺省值	说明
safe_copy_count_in_init	2	Request copy count in init. Range: (0,3)
safe_lost_one_time	3600s	Safe duration while lost one copy. You are advised to use the default value.
safe_wait_init_time	60s	Time interval for build root table.
schema_filename	etc/schema.ini	Schame file name.
slave_register_timeout	3s	Slave register process timeout time. You are advised to use the default value.
tablet_migrate_disabling_period	60s	Chunk Server can participate in balance after regist. You are advised to use the default value.
tablet_replicas_num	3	Tablet replicas num.
ups_lease_reserved_time	8500ms	Update Server lease reserved time. You are advised to use the default value.

参数	缺省值	说明
ups_lease_time	9s	Update Server lease time. You are advised to use the default value.
ups_renew_reserved_time	7770ms	Update Server renew reserved time. You are advised to use the default value.
ups_waiting_register_time	15s	Root Server select master Update Server, should wait the time after first Update Server regist.
vip_check_period	500ms	Virtual IP check period. You are advised to use the default value.
write_queue_size	100	Write queue size. Range: [10,100000]

7.4.2 Update Server 配置参数

Update Server 配置参数说明如表 7-20 所示。

表 7-20 Update Server 配置参数

参数	缺省值	说明
active_mem_limit	0	Active memtable memory limit.
allow_write_without_token	True	allow write without token.

参数	缺省值	说明
blockcache_size	0	Block cache size.
blockindex_cache_size	0	Block index cache size.
commit_log_dir	data/ups_commitlog	Update Server commit log directory.
commit_log_size	64MB	Commit log size.
consistency_type	2	Consistency type of log-sync. 1: strong 2: normal 3: weak
devname	bond0	Listen device.
dir_regex	^store[0-9]+$	Store regex to find store directory. You are advised to use the default value.
disk_warn_threshold	5ms	Disk warn threshold.
fetch_log_wait_time	500ms	Fetch log retry wait time. You are advised to use the default value.
fetch_schema_timeout	3s	Active fetch shema timeout. You are advised to use the default value.

参数	缺省值	说明
fetch_schema_times	10	Active fetch schema try times if fail. You are advised to use the default value.
inner_port	2710	Inner port for daily merge. You are advised to use the default value. Range: (1024,65536)
io_thread_count	3	I/O thread number for libeasy.
keep_alive_timeout	5s	Keep alive timeout. You are advised to use the default value.
lease_queue_size	100	Lease queue size.
lease_timeout_in_advance	500ms	Lease timeout in advance. You are advised to use the default value.
log_cache_block_size	32MB	Size of per-block of log cache.
log_cache_n_block	4	Number of blocks of log cache.
log_queue_size	100	Log queue size.

参数	缺省值	说明
log_sync_delay_warn_report_interval	10s	Commit log delay alarm given interval.
log_sync_delay_warn_time_threshold	500ms	Commit log delay beyond this value beyond this valud between master and slave Update Server will give an alarm.
log_sync_retry_times	2	Log sync retry times. You are advised to use the default value.
log_sync_timeout	500ms	Slave sync log timeout.
log_sync_type	1	Sync log to disk.
low_priv_adjust_flag	True	True: Auto adjust the probalitity to deal with Low priority task. False: Not Auto adjust the probalitity to deal with Low priority task
low_priv_cur_percent	10	Current low priority process probability. Range: [0,100]

参数	缺省值	说明
low_priv_network_lower_limit	30MB	Increase 1% probability to process low priority if low priority request network band less than this value and 'low_priv_adjust_flag' is True.
low_priv_network_upper_limit	80MB	Decrease 1% probability to process low priority if low priority request network band beyond this value and 'low_priv_adjust_flag' is True.
lsync_fetch_timeout	5s	Fetch commit log timeout from lsync or master Update Server.
lsync_ip	0.0.0.0	Lsync IP address.
lsync_port	3000	Lsync listen port. Range: (1024,65536)
major_freeze_duty_time	Disable, OB_CONFIG_DYNAMIC	Major freeze duty time.
max_n_lagged_log_allowed	10000	Commit log laged count beyond this value beyond this valud between master and slave ups will give an alarm.

参数	缺省值	说明
max_row_cell_num	256	Compact cell when cell of row beyond this valud.
memtable_hash_buckets_size	0	Number of hash index buckets. You are advised to use the default value.
min_major_freeze_interval	1s	Minimal time to generate major freeze version.
minor_num_limit	0	Using major freeze instead if number minor version greater or equal to this value.
net_warn_threshold	5ms	Net worn threshold.
packet_max_wait_time	10s	Default RPC(Remote Process Call) timeout if not timeout specified. You are advised to use the default value.
port	2700	Update Server's port.
raid_regex	^raid[0-9]+$	Raid regex to find raid directory. You are advised to use the default value.

参数	缺省值	说明
read_queue_size	1000	Read queue size.
read_thread_count	4	Read thread number.
real_time_slave	True	True: The server is a realtime slave Update Server. False: The server is not a realtime slave Update Server.
refresh_lsync_addr_interval	60s	Interval of slave to refresh lsyncserver-address.
register_timeout	3s	Register to root server timeout. You are advised to use the default value.
register_times	10	Register to root server try times if fail. You are advised to use the default value.
replay_checksum_flag	True	True: Checksum when replay. False: Not checksum when replay.
replay_log_buf_size	10GB	Replay log buffer size. You are advised to use the default value.

参数	缺省值	说明
replay_queue_len	10000	Replay queue size. You are advised to use the default value.
replay_wait_time	100ms	Replay retry wait time. You are advised to use the default value.
replay_worker_num	0	Replay worker number.
resp_root_timeout	1s	Report frozen version to root server timeout. You are advised to use the default value.
resp_root_times	20	Report frozen version to root server try times if fail. You are advised to use the default value.
retry_times	3	Retry times if failed.
root_server_ip	-	Root server' IP.
root_server_port	3500	Root server listen port. Range: (1024,65535)
slave_sync_sstable_num	1	Not used now.

参数	缺省值	说明
sstable_block_size	4K	Sstable block size.
sstable_compressor_name	none	Sstable compressor name.
sstable_time_limit	7d	Remove from memory and dump to trash directory if sstable stay in memory such time.
state_check_period	500ms	Interval of slave to check sync-stat. You are advised to use the default value.
store_queue_size	100	Store queue site.
store_root	data/ups_data	Update Server data directory. You are advised to use the default value.
store_thread_count	3	Store thread number.
table_available_error_size	0	Force drop frozen table and give an alarm if available table memory less than this value. You are advised to use the default value.

参数	缺省值	说明
table_available_warn_size	0	Try drop frozen table if available table memory less than this value. You are advised to use the default value.
table_memory_limit	0	Table memory limit.
total_memory_limit	0	Total memory limit.
trans_proc_time_warn	1s	If master process batch or slave write local log beyond this value, give an alarm. You are advised to use the default value.
trans_thread_num	0	Number of thread to process read/write transaction.
using_hash_index	True	True: Using hash index. False: Not using hash index.
using_memtable_bloomfilter	False	True: Using memetable bloomfilter. False: Not using memtable bloomfilter.

参数	缺省值	说明
using_static_cm_column_id	False	True: Should treat 2 and 3 as create_time and modify_time column id. False: Not treat 2 and 3 as create_time and modify_time column id.
wait_slave_sync_time	100ms	Wait slave sync time. You are advised to use the default value.
wait_slave_sync_type	0	0: response master Update Server before replay. 1: Response master Update Server after replay before sync to disk. 2: Response master Update Server after sync to disk.
warm_up_time	10m	Sstable warm up time. Range: [10s,1800s]
write_queue_size	1000	Write queue size.

参数	缺省值	说明
write_sstable_use_dio	True	True: Write sstable use DIO(Direct Input-Output). False: Write sstable not use DIO
write_thread_batch_num	1024	Max wirte task count for batch. Range: [1,∞)

7.4.3 Merge Server 配置参数

Merge Server 配置参数说明如表 7-21 所示。

表 7-21 Merge Server 配置参数

参数	缺省值	说明
allow_return_uncomplete_result	False	True: Not allow return uncomplete result. False: Allow return uncomplete result.
devname	bond0	Listen device.
frozen_version_timeout	600s	Update Server frozen version cache tiemout.
intermediate_buffer_size	8MB	Intermediate buffer size to store one packet, 4 times network packet size (2M).
io_thread_count	1	I/O thread count for libeasy. Range: [1,∞)

参数	缺省值	说明
lease_check_interval	6s	Lease check interval. You are advised to use the default value.
location_cache_size	32MB	Location cache size.
location_cache_timeout	600s	Location cache timeout. You are advised to use the default value.
log_interval_count	100	Legacy param, used for OB0.3 Only.
max_get_rows_per_subreq	20	Row count to split to cs when using multi-get, 0 means no split. Range: [0,∞)
max_parellel_count	16	Max parellel sub request to chunkservers for one request. Range: [1,∞)
max_req_process_time	15s	Max process time for each request. You are advised to use the default value.
memory_size_limit_percentage	40	Max percentage of totoal physical memory ms can use. Range: (0,100]

参数	缺省值	说明
monitor_interval	600s	Execute monitor task once every monitor_interval.
network_timeout	2s	Timeout when communication with other server.
obmysql_io_thread_count	1	Obmysql I/O thread count for libeasy. Range: [1,∞)
obmysql_port	3100	Obmysql listen port. Range: (1024,65536)
obmysql_task_queue_size	10000	Obmysql task queue size. Range: [1,∞)
obmysql_work_thread_count	50	Obmysql I/O thread count for doing sql task. Range: [1,∞)
port	2800	Merge Server's port.
query_cache_size	0	Query cache size, 0 means disabled. Range: [1,∞)
reserve_get_param_count	3	Legacy param, used for OB0.3 Only. Range: [1,∞)
retry_times	3	Retrytimes if failed.

参数	缺省值	说明
root_server_ip	-	Root server' IP.
root_server_port	3500	Root server listen port. Range: (1024,65535)
slow_query_threshold	100ms	Query time beyond this value will be treat as slow query.
task_left_time	100ms	Task left time for drop ahead. You are advised to use the default value.
task_queue_size	10000	Task queue size. You are advised to use the default value. Range: [1,∞)
task_thread_count	10	Task thread number.
timeout_percent	70	Max Chunk Server timeout to ms timeout, used by Merge Server retry. Range: [10,80]
use_new_balance_method	True	True: Use new balance method. False:Not use new balance method

7.4.4 Chunk Server 配置参数

Chunk Server 配置参数说明如表 7-22 所示。

表 **7-22 Chunk Server** 配置参数

参数	缺省值	说明
appname	-	Application name.
block_cache_size	1GB	Block cache size. Range: (0,∞)
block_index_cache_size	512MB	Block index cache size. Range: (0,∞)
bypass_sstable_loader_thread_num	0	Bypass sstable loead thread number. Range: [0,10]
check_compress_lib	snappy_1.0: none: lzo_1.0	Check compress lib as cs start.
compactsstable_cache_size	0	Compacet sstable cache size.
compactsstable_cache_thread_num	0	Compacet sstable cache thread number. Range: (0,∞)
datadir	./data	Sstable data path.
devname	bond0	Listen device.

参数	缺省值	说明
each_tablet_sync_meta	True	True: Sync tablet image to index file after merging each tablet. False: Sync tablet image to index file once every day or when killling the Chunk Server.
fetch_ups_interval	5s	Fetch Update Server list interval.
file_info_cache_num	4096	File info cache number. Range: (0,∞)
groupby_mem_size	8MB	Maxmum memory used in groupby operator.
io_thread_count	4	I/O thread number for libeasy. Range: [1,∞)
join_batch_count	3000	Join row count per round. Range: (0,∞)
join_cache_size	512MB	Join cache size.
lazy_load_sstable	True	True: Lazy load sstable to speed up chunkserver start. False: Not load sstable while chunkserver start.

参数	缺省值	说明
lease_check_interval	5s	Lease check interval, shouldn't change. You are advised to use the default value. Range: [5s,5s]
max_groupby_mem_size	16MB	Clear memory over this size after groupby.
max_merge_mem_size	16MB	Clear memory over this size after each sub merge.
max_merge_thread_num	10	Max merge thread number. Range: [1,32]
max_migrate_task_count	2	Max migrate task number. Range: [1,∞)
max_version_gap	3	Use to judge if the seving version is too old, maybe need not to merge. Range: [1,∞)
merge_adjust_ratio	80	When the load is greater than this ratio of merge_load_high, slow down daily merge.

参数	缺省值	说明
merge_delay_for_lsync	5s	Sleep time wait for Update Server synchronise frozen version if merge should read slave Update Server. Range: (0,∞)
merge_delay_interval	600s	Sleep time before start merge. Range: (0,∞)
merge_highload_sleep_time	2s	Sleep time if system load beyond 'merge_threashold_load_high' in merge check.
merge_mem_limit	64MB	Memory usage to merge for each thread.
merge_mem_size	8MB	Memory for each sub merge round, finish that round if cell array oversize.
merge_migrate_concurrency	False	True: Allow doing merge and migrate concurrently False: Not allow doing merge and migrate concurrently.
merge_pause_row_count	2000	Merge check after how many rows.

参数	缺省值	说明
merge_pause_sleep_time	0	Sleep time for each merge check.
merge_scan_use_preread	True	True: prepread sstable when doing daily merge. False: Not prepread sstable when doing daily merge.
merge_thread_per_disk	1	Merge thread per disk, increase the number will reduce daily merge time but increase response time. Range: [1,∞)
merge_threshold_load_high	16	Suspend some merge threads if system load beyond this value. Range: [1,∞)
merge_threshold_request_high	3000	Suspend some merge threads if get/scan number beyond this value. Range: [1,∞)
merge_timeout	10s	Fetch ups data timeout in merge. Range: (0,∞)
merge_write_sstable_version	2	Sstable version. Range: [1,∞)

参数	缺省值	说明
migrate_band_limit_per_second	50MB	Network band limit for migration.
min_drop_cache_wait_time	300s	Waiting time before drop previous version cache after merge done.
min_merge_interval	10s	Minimal merge interval between tow merges.
network_timeout	3s	Timeout when communication with other server.
over_size_percent_to_split	50	Over size percent to split sstable. Range: (0,∞)
port	2600	Chunk Server's port.
retry_times	3	Retry times if failed.
root_server_ip	-	Root server' IP.
root_server_port	3500	Root server listen port. Range:(1024,65535)
slow_query_warn_time	500ms	Beyond this value will treated as slow query.
sstable_row_cache_size	2GB	Sstable row cache size. Range: (0,∞)

参数	缺省值	说明
switch_cache_after_merge	False	True: Switch cache after merge. False: Not switch cache after merge.
task_left_time	300ms	Time left to Merge Server, drop ahead if left time less than this value.
task_queue_size	10000	Task queue size. Range: [1000,∞)
task_thread_count	20	Task thread number. Range: [1,∞)
unmerge_if_unchanged	True	True: Not mege sstable. False: Merge sstable.
ups_blacklist_timeout	5s	Remove Update Server if it stay in blacklist over this time.
ups_fail_count	100	Put Update Server to blacklist if fail count beyond this value. Range: [1,∞)
write_sstable_use_dio	True	True: Write sstable use DIO. False: Write sstable not use DIO.